Methodology for Formal Verification of Routing Protocols for Ad Hoc Wireless Networks

D. Câmara
Department of Computer Science
Federal University of Minas Gerais
30123-970 Belo Horizonte, MG, Brazil

A. A. F. Loureiro
Department of Computer Science
Federal University of Minas Gerais
30123-970 Belo Horizonte, MG, Brazil

F. Filali
Department of Mobile Communications
Institut Eurécom
06904 Sophia-Antipolis, France

***Abstract* – This paper describes a technique to apply formal methods to verify protocols for mobile ad hoc networks. In contrast to other related proposals, our solution does not attempt to model any particular network configuration. Instead, our solution focuses on the possible implications caused by network configurations to the behavior of a routing protocol for MANETs. Following this strategy we were able to find design errors in some well established protocols. The proposed technique uses formal verification, more specifically model checking, to detect, in a simple way, problems such as routing loops, delivery message failures and errors in the protocol state machine.**

I. Introduction

Routing protocols play an important role in computer networks once they are responsible for redirecting packets and discovering routes through the computer network. The correctness of these protocols is crucial in order to have more sable and trustful networks. Unfortunately the development of routing protocols, mainly for wireless networks, is a complex and error prone task. This occurs, not only because the distributed nature of the problem and its inherent complexity, but also because the lack of good tools to help the protocol designer on its task. Formal methods, especially formal verification, can help the protocol designers to decrease the development time , find design errors and quick validate solutions for the encountered errors. Thus the use of such tools improves the final quality of the protocols.

In the last years, many proposals have used formal verification to validate routing protocols for wireless ad hoc networks. However, such techniques differ from the one presented here since they are: normally complex and unsuitable to model mobility; some times applicable to one single problem, or protocol; and typically such techniques propose proofs based on particular scenarios. This work, on the other hand, presents a simple, and more important, topology independent approach.

The main contributions of this work are as follows: first it presents new design failures found in some well known and established routing protocols for wireless networks. Second, it presents a way to make the verification topology independent, avoiding a proof based on a particular scenario. Third, the method presents a way to safely model flooding based protocols, avoiding the combinatorial state space explosion problem . Fourth, to decrease the protocol verification complexity, the method proposes divide the verification into internal and external protocol behavior. Finally, this work presents a simple, yet powerful, method to apply formal verification to validate routing protocols for mobile ad hoc networks. To validate our methodology, three different and well established routing protocols for MANETs were verified. After applying our technique, we were able to find design flaws in all evaluated protocols.

This work is organized as follows. The next section presents the related work focusing on the strengths and weaknesses of the different proposals. Section 3 presents the proposed methodology to model and verify protocols for mobile ad hoc networks. In Section 4, we apply our technique to three different protocols and present the flaws discovered with the application of our method. Finally, Section 5 presents the conclusions and future extensions of this work.

II. Related Work

Formal methods, in general, refer to mathematically based techniques used in specification, development and verification of software and hardware systems. The use of formal methods increases the rigor on development of computational systems, leading to more reliable ones. Because the mathematics involved in the process, people tend to believe that the use of formal methods, manly formal verification, worthies only for safety critical systems. However, formal methods may help the development of any system and the mathematics involved on it is, in general, quite easy and straightforward .

This work focus on automated model checking techniques. Model checking is a method to verify whether a formally modeled system satisfies a given logic specification . In automated model checking the prover tool often varies some properties using an exhaustive search of all possible states that the model could reach during its execution. On the other hand, in our technique we propose to identify all *possible relations* available on the model, not necessary all protocol states.

In and Obradovic et. al show how to use the theorem prover HOL and the model checker SPIN to prove key properties of distance vector routing protocols. The technique focus mainly on distance vector algorithms. The main disadvantage of this work is the intense user interaction . HOL is a semi automatic theorem prover that needs the user to guide it. Another problem is the complexity in defining the theorems and lemmas to perform the real proof.

In , Wibling et al. use model checking to verify the Lightweight Underlay Network Ad-hoc Routing (LUNAR) Protocol. They use SPIN to verify the data and control aspects of the LUNAR protocol and the UPPAAL tool to verify the protocol timing properties. A possible drawback is that the authors only verify LUNAR, which was designed by the same group. Furthermore, the work is also based on some strong assumptions: only bidirectional links are allowed, messages must be delivered in order, and each node in the network can only receive and handle one message at a time. Such assumptions, in some cases, may even prevent the whole protocol verification, if it is based on any of these points.

Chiyangwa and Kwiatkowska focus their work on the timing aspects of AODV using UPPAAL. They build a timed automata model and evaluate the effects of the standard protocol parameters on the timing behavior of AODV. In that work, they evaluate a linear topology in which the source is node *0* and the destination is node *n-1*. All other nodes involved are sequentially placed between the source and the destination. The work focuses on this particular topology because it intends to evaluate timing aspects of the routing discovery problem and to find the maximum possible network diameter. The work reaches its purpose, but it only verifies timing aspects of the protocol and does not consider qualitative aspects, such as loops and other routing problems.

In Yuan et al. illustrate the dynamic operations of a MANET using Coloured Petri Nets. They show a simple, yet powerful, way to model the dynamic topology changes of ad hoc networks with Colored Petri Nets. Its great strength is the simplicity and elegance of the model they propose to handle mobility. However, because of the simplifications in the modeling process, the work does not really handle the process of sending messages. Thus there are no differences between full and incremental routing table updates, for example. This simplification may hide important errors that are not verified. The technique also does not allow two different nodes to receive and process, simultaneously, broadcast messages. In this case errors caused by concurrent sending/receiving messages are not be detected.

III. The Methodology

This work is an improvement over the one presented in , and it is a step-by-step procedure to verify routing protocols for wireless ad hoc networks. In contrast to the work presented in [2], our method focuses mainly in qualitative aspects rather than quantitative ones. We are interested in identify routing loops, packet delivery failures, unexpected reception of messages and "pathological'' cases not treated in the original protocol specifications. However, quantitative aspects, such as max/min number of messages exchanged among nodes, behavior in a particular topology or timing aspects, for now, are not the focus of the method.

Typical formal verification approaches applied to routing protocols for MANETs, such as Wibling et al. , and Chiyangwa and Kwiatkowska , use either a specific network configuration or a given number of nodes on the verification. The problem with these approaches is that mobile ad hoc and sensor networks are dynamic systems. Therefore, the correctness proof of a particular configuration does not guarantee the correctness of the protocol with respect to other configurations.

This work is grounded on a complete different principle. It does not model any particular network configuration. Instead of that, it proposes that one should model all the possible *implications* caused by network configurations to the behavior of the routing protocol for MANETs. In other words, the verification should model all the possible *relations* among the nodes. This is a key observation in our technique.

1. Ground Principles

To decrease the complexity of the models and avoid the combinatorial explosion problem this work proposes that the verification must follow some principles. They are: topology abstraction, node position and lower layer services.

Topology abstraction: any proof that takes into account a specific topology just proved that the protocol works for that particular topology. So, instead of enumerate the infinite possible topologies, it is better to avoid it. In order to do that the methodology proposes to use three *kinds* of nodes, namely, source, destination and intermediate nodes. The intermediate node, in truth, represents not one node, but a set of nodes and all possible effects of interconnections among them. For example, a node with a neighbor and a packet transmitted successfully, a node with a neighbor and a lost packet, and so on. With this simple idea the topology becomes irrelevant once the cloud of intermediate nodes will model the possible *relations* not the topologies, in truth the method does not rely in any particular topology what makes of it topology independent. This greatly decreases the complexity and, in consequence, the effort required to formally verify the dynamics of routing protocols.

Node position: position awareness is one of the criteria for classifying routing algorithms. However, with the topology abstraction, the node position becomes irrelevant. This stands even for position aware algorithms, since we are concerned with the possible relations, not specific configurations.

Lower layers services: once they are not the object of study, the services provided by the lower layers should be modeled as available and trustable, unless some cross layer aspect is crucial for the protocol validation. In this case, the verification, for this cross layer aspect, should be done apart.

2. *Modeling*

Communicating channel: the communicating channel should be available and common to every node in the network, in a random way. This means that any node, or even no one at all, may receive the packet. However, the way the protocol is designed must determine if the packet will actually reach the destination or not, regardless the way the channel is modeled.

Flooding representation: with this communication channel two messages can represent all existing relations in a flooding. However, when verifying a protocol, the designer must make sure that all *relations* are reachable.

Mobility: from the node point of view, the main consequence of the mobility is the occurrence of broken links. So, as the model represents all possible relations among nodes, including broken links, the mobility is also modeled.

The network: more important than to identify all possible topologies is to identify all possible *effects* of different topologies and node states to the routing protocol. For instance, what can happen to a message after it is received by a node? Some possible scenarios are: it can be lost because of a node failure, it can be transmitted to a neighbor node with a checksum error, it can be transmitted successfully, and so on.

Internal and external behavior: the protocol should be divided into internal and external behavior, and each part verified separately. Internal behavior refers to how the protocol handles data and controls messages internally to the node. In other words, the actions the node implements when it receives, or sends, a specific message. The external behavior refers to how the whole network reacts to the messages. Both models should be independent and modeled in such way that the internal behavior could act as a "procedure" of the external behavior. This decreases the model complexity without compromising the verification correctness because all aspects could be easily integrated. Both behaviors start, in general, with a packet and its relations. The initialization of the packet data and its relations should be random to guarantee the coverage of all relations.

Information modeling: every information regarding the verified protocol should be modeled as a variable and, as far as possible, randomly initialized (e.g., package type, packet time-to-live (TTL), and table exchange trigger). When possible, such information should also be modeled with boolean variables. For example, if the packet has a TTL, not all its values need to be verified: if the TTL was reached, or not, is normally enough. In PROMELA this can be represented as:

```
bool ttl;
 if
     :: (1) -> ttl = 1
     :: (1) -> ttl = 0
 fi;
```

Simplifications: to avoid the combinatorial state space explosion a protocol should be simplified as much as possible, while it does not compromise the verification results. Once over simplification can often lead to wrong conclusions. As an example of a simplification, suppose a protocol that uses the Dijkstra or Floyd protocol to find the minimum path. For the verification purpose, it may be enough to model the shortest path as a Boolean variable, either the protocol finds the path or not. When verifying, the designer should be, normally, more concerned about his own algorithm than the shortest path one.

Abstraction: the protocol model should start with a simple abstraction and increase its complexity appropriately. With this approach, basic problems can be identified earlier and possible solutions can be quickly validated. This also allows the protocol designer to stop verifying the protocol whenever it reaches a goal or a reasonable complexity.

Analysis: every time a property is verified and the tool presents a response scenario, the designer must analyze whether it is a fault on the protocol or on the model. For example, simplifications can introduce errors in the model that are not actually possible in the real world. This is a crucial step and, unfortunately, it can not be done automatically.

In the following, we present a simplified description of the verification methodology in an algorithmic form.

ALGORITHM
VERIFICATION METHODOLOGY STEPS

```
1. Acquire needed information to model the protocol;
2. Create a detailed pseudo-code or finite state machine of the protocol;
3. Compare carefully all cases described in the protocol with the pseudo code
   and verify if they are consistent, if not repeat the previous steps;
4. Create a table with all kinds of packets and the nodes that can generate
   them (source, intermediate and destination node);
   4.1. Specify the semantics of the packets to each node;
5. Divide the protocol into internal and external behaviors
     ▪ Internal behavior: describes the message flows and behaviors for
       the node;
     ▪ External behavior: describes the behaviors related to the node
       interactions;
   5.1. Understand each aspect of the protocol, create an algorithm or an
        state machine representation to understand it better;
6. Model the External vs. Internal interactions
     ▪ The internal behavior should be modeled as if it was a routine
       call. In this way the external behavior becomes independent of the
       internal behavior. Ideally, the external and internal behaviors
       should be independent;
7. Start with a simple model and continuously increase the model complexity
   7.1. For each error found
        7.1.1. Verify whether the error is due to a protocol failure or a
               modeling failure;
        7.1.2. Find a solution for the problem;
        7.1.3. Model the solution;
        7.1.4. Test the solution;
   7.2. Increase the model complexity;
8. Identify and isolate verified procedures to be used in other protocols.
```

IV. METHODOLOGY APPLIED

To validate the methodology, we use three different routing protocols for MANETs: LAR , DREAM and OLSR . The first two algorithms are geographic routing protocols and the last one is a link state based protocol. Such algorithms were carefully chosen to show the usefulness of the methodology. The first two algorithms are well-established geographic routing algorithms with some well known flows. The idea behind this is to know whether the methodology can be applied to geographic routing protocols, ignoring the node position, and whether the methodology is able to, at least, detect the known flows of such algorithms. Another point observed is that they are flooding-based protocols, and we wanted to make sure our technique really works with this class of algorithms. On the other hand, OLSR is a well known and cited algorithm, which became an RFC , and uses a completely different routing approach.

Without exception all evaluated protocols, LAR1, LAR2, DREAM and OLSR, presented problems. Some of them are well known, such as the inability of both LAR and DREAM to find an existing route. However, others such as the presence of a loop on both DREAM and LAR2, in the best of our knowledge were not previously known for these algorithms.

A. *Evaluated protocols*

The Location-Aided Routing (LAR) routing protocol has two different variants, LAR1 and LAR2. Both are geographic routing algorithms but working in a quite different way. LAR1 defines a rectangular requesting zone defined by the two extreme points of the following diagonal: the sender position and the old destination position including a circular expected zone of this node. Inside the requesting zone the packets are flooded to reach the destination. In LAR2, instead of a rectangular expected zone, every node, when receives a message, calculates its own distance from the destination and verifies if such distance is greater than the one from the previous node, considering a threshold. If this distance is greater, the packet is discarded, if not the packet is forwarded.

The Distance Routing Effect Algorithm for Mobility (DREAM) is based on two simple observations, distance effect and mobility rate. When a node needs to send a message to any other node, it verifies the position, velocity and information time of the destination node on its own routing table. With these the source node estimates the area where the node is and calculates the angle , which defines the destination node expected reachable zone. With this expected zone the source defines a triangular region between its position and the tangents to the expected zone. Then it sends a unicast message to the nodes within that region, which in their turn repeat the same procedure until the message reaches the destination.

The Optimized Link State Routing (OLSR) is an optimized version of the standard link state routing algorithm. OLSR periodically exchanges messages to update its routing table. Instead of using a complete path, the protocol uses a hop-by-hop approach that allows each node to use the most recent information to forward the packet. To decrease the number of control messages OLSR uses Multi Point Relays (MPRs). Nodes chosen as MPRs are the minimal set of nodes required to send a message to all two-hop distant neighbors, and only the MPRs nodes can forward messages. This guarantees entire coverage of those two hops nodes and with the minimum number of sent messages.

B. *Modelling*

In this work, we created all models in PROMELA and verified them using SPIN . However, any other model checker could be used. SPIN is a generic verification system largely used to model and verify distributed systems. SPIN accepts design specifications written in PROMELA and properties to verify in Linear Temporal Logic (LTL).

The properties verified in this work where loop occurrence, valuable packet dropped and packet not delivered. Each one of such properties were modeled as boolean variables and the occurrence or not of the property was the LTL formula verified. The scenarios were small and occupied from 1.4 to 1.6 MB of space for each property verified. All verifications used the default SPIN hash table state size, which is 2^{18}. The stored or reachable states for the protocols, with the verified properties, varied from 91 to 2489. The maximum longest depth-first search path for the verified properties varied from 17 to 90 steps. These data are indicative of how tractable the models became.

C. *Results*

Among the design errors found using the methodology we report: (*i*) fail to deliver messages in LAR1, LAR2 and DREAM, even though all three protocols use controlled flooding to deliver messages, (*ii*) loop scenarios in LAR1, LAR2 and DREAM, somehow unexpected for geographic routing algorithms; (*iii*) fail to deliver messages in OLSR, when a message arrives during a routing table recalculation; (*iv*) discard newer table information in OLSR and (*v*) control messages discarded in OLSR.

Fig. 1 shows a scenario where LAR2 fails to deliver a message, although there is a path between source and destination. This occurs if any node in the path is farther from the destination than the previous node. This fail can occur also in DREAM, Fig. 3, if the path is not inside the expected zone. In LAR1 the delivery failure may occur if there is a path outside the expected zone and no path inside it. This is interesting because even doing a flooding there is no guarantees that LAR or DREAM will effectively deliver a message. Another identified situation where LAR1 may fail delivering a message is when the expected zone and the origin node are aligned. When this occours few, or no nodes at all, are found inside the expected zone, and, thus, the packet is lost.

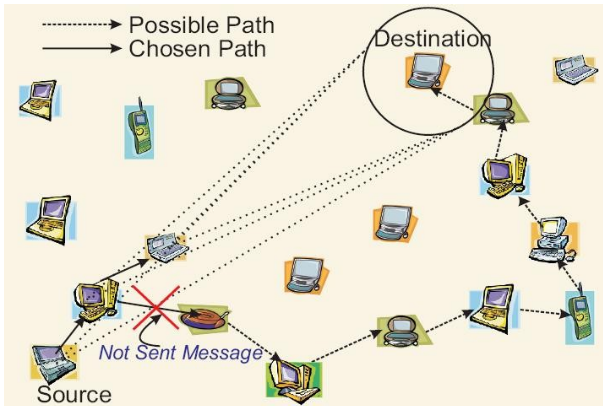

Figure 1. Delivery failure in Lar2 when a path it is available, problem detected by the methodology.

In a broader point of view all three protocols may fail delivering messages because route concavity. In other words, if the *only* viable route passes through a node and this node is far from the destination than the previous one this path is discarded and the message will never reach the destination.

Fig. 2 shows the loop scenario in LAR2. This situation occurs when all nodes are at the same distance from the destination, considering as a distance threshold. In this case the message will be forwarded from node to node indefinitely.

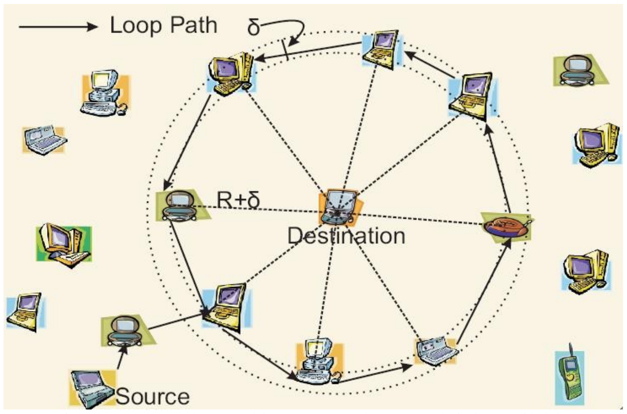

Figure 2. Loop detected in LAR2, when a path it is available, and all nodes are in the re gion in a near circle.

The loop on LAR1 occurs whenever the time during which a packet is retransmitted inside the requested zone is greater than the time nodes keep track of the transmitted packets.

The loop on DREAM, Fig. 3, can occur if the search angle is greater than 90^o. The path may not converge and loops are possible. This is somewhat unexpected as DREAM authors claim their algorithm being loop free . This just reinforces the need of formal verification for routing protocols.

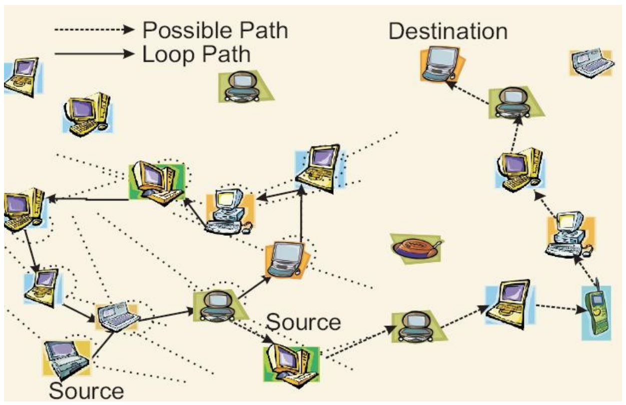

Figure 3. Loop and delivery failure discovered in DREAM when a series of nodes are in the angle of dissemination is big enough.

OLSR may fails to deliver messages, when these arrive during a routing table recalculation. As the protocol is described in , all table entries are erased during the routing table recalculation. If a packet arrives exactly at this time it may be discarded. The second problem found with OLSR is that it does not control counter overflow, so whenever a counter overflow occurs, the older information is kept on the routing tables instead of newer ones. Other problem that occurs in OLSR is that when a message arrives in a node, just after the link is marked as unidirectional instead bidirectional, the control messages may be discarded and not all two hop neighbors may receive it. Again, in this case authors argue that the MPR nodes are enough to guarantee that all two-hop neighbors will receive the control messages, once they represent the minimum set to cover all two-hop away nodes.

V. Conclusions

The method presented here showed to be robust and useful to confirm the existence of problems in algorithms and even to find new ones. With it, we were also able to verify flooding based protocols, which, in general, become a problem for other techniques because the state space explosion . The topology abstraction, channel and information modeling showed to be an effective and reliable form to build verification models for routing algorithms.

However, the current version of the method does not help determining the protocol limits. Other problem is that the error scenarios must to be manually evaluated by the designer who also needs to determine the sources of such errors. Currently, we are working on extensions of this technique to other protocols and building a library of verified procedures that can be used to create more reliable protocols in the future. The protocol verification built on top of such procedures, may simplify the work of protocol designers and grant more reliable protocols in the future.

References

[1] O. Wibling, J. Parrow, A. Pears, Ad Hoc Routing Protocol Verification Through Broadcast Abstraction, 25th IFIP FORTE, Taiwan, 2005
[2] Sibusisiwe Chiyangwa and Marta Kwiatkowska. A timing analysis of AODV, 7th IFIP FMOODS, June 2005.

[3] D, Câmara, C. F. Santos, A. A . F. Loureiro, Formal Verification of Routing Protocols for Ad hoc Networks, Brazilian Symposium on Computer Networks, SC, Brazil, 2001. (In Portuguese)

[4] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward, A Distance Routing Effect Algorithm For Mobility, MobiCom'98, Dallas, TX, 1998

[5] Y. Ko and N. H. Vaidya, Location-Aided Routing (LAR) Mobile Ad Hoc Networks, MobiCom'98, Dallas, TX, 1998

[6] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, L. Viennot, Optimized Link State Routing Protocol for Ad Hoc Networks, IEEE INMIC Pakistan 2001

[7] E. M. Clarke, O. Grumberg, and D. A. Peled. Model Checking. MIT Press, 1999

[8] A. Hall, Seven Myths of Formal Methods, IEEE Software, Sept 1990

[9] O. Wibling, Ad Hoc Routing Protocol Validation, Licentiate Thesis 2005-004, Dept of Info Technology, Uppsala University, Sweden, 2005

[10] C. Yuan, J. Billington, An Abstract Model of Routing in Mobile Ad Hoc Networks, Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark, 2005

[11] K. Bhargavan, D. Obradovic, C. A. Gunter, Formal verification of standards for distance vector routing protocols, Journal of the ACM, 538-576, Volume 49, Number 4, July 2002

[12] D. Obradovic, Formal Analysis of Convergence of Routing Protocols, Ph.D. Thesis Proposal, Department of Computer and Information Science, University of Pennsylvania, November 28, 2000

[13] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, A distance routing effect algorithm for mobility (DREAM),in ACM/IEEE Mobicom'98, pages 76 - 84.

[14] T. Clausen, P. Jacquet, Optimized Link State Routing Protocol (OLSR), Request for Comments: 3626, October 2003

[15] G. J. Holzmann, The model checker SPIN. IEEE Trans. on Software Eng., 23(5):279--295, May 1997

[16] F. J. Lin, P. M. Chu, M. T. Liu, Protocol verification using reachability analysis: the state space explosion problem and relief strategies, SIGCOMM '87, ACM Press, 1988